R Notebook

Part 1: Create a function that identifies all candidate guide (protospacer) sequences from a given genomic region.

```
################################################################################################
##  Function protospacer will identify the PAM sequences of the input DNA sequence and will
##  retrieve the protospacer sequence based on the given length (20bp)
##  - When a sequence has only one NGG present
##  - When a sequence has more than one NGG present including the presence of multiple GGG
##  - When a sequence has no NGG present
###############################################################################################

library(stringr)
s1 <- "TGATCTACTAGAGACTACTAACGGGGATACATAG" #example sequence
s2 <- "ATACTAGTACATACTAACTCTAACTGGATCGATAAA"
s3 <- "ATCGTGCCTTAAGCTA"
l <- 20
p <-  "NGG"
###############################################################################################
## When a sequence has only one NGG present, we identify the [A|T|C]GG in a sequence and count
## 20bp backwards and print the sequence
###############################################################################################

protospacer <- function(sequence, length)
{
  locate <- as.data.frame(str_locate_all(sequence, "GG"))
  n <- nrow(locate)

  if ((nrow(locate) == 1) && (locate$start >= 22) == TRUE)
    {
      substring1 <- str_sub(sequence, start = 1, end = locate$start[1] - 2)
      rev_string <- stringi::stri_reverse(substring1)
      sequence1   <- str_sub(rev_string, start = 1, end = l)
      rev_string <- stringi::stri_reverse(sequence1)
      return(rev_string)
    }

################################################################################################
## When a sequence has more than one NGG present, we identify the positions of each of 'GG'
## present in the sequence by using str_locate_all function and print the sequences 20bp before ## by u
################################################################################################

list_val <- c()
if (nrow(locate) >= 2)
{
  for (i in range(1, n))
  {
    if (locate$start[i] >= 22)
```

```r
    {
      if (grepl("GG", str_sub(sequence, start = locate$start[i], end = locate$end[i])) == TRUE)
      {
        substring2  <- str_sub(sequence, start = 1, end = locate$start[i] - 2)
        rev_string2 <- stringi::stri_reverse(substring2)
        sequence2   <- str_sub(rev_string2, start = 1, end = l)
        rev_string2 <- stringi::stri_reverse(sequence2)
        if ((is.null(list_val[i]) ||  (is.na(list_val[i])) == TRUE))
        {
          list_val[i] <- rev_string2
        }
        else
        {
          list_val[i + 1] <- rev_string2
        }

      }
      if (i != n)
      {
         if (grepl("GG", str_sub(sequence, start = locate$end[i], end = locate$start[i + 1])) == TRUE)
         {
           substring3  <- str_sub(sequence, start = 1, end = locate$end[i] - 2)
           rev_string3 <- stringi::stri_reverse(substring3)
           sequence3   <- str_sub(rev_string3, start = 1, end = l)
           rev_string3 <- stringi::stri_reverse(sequence3)
           if ((is.null(list_val[i + 1]) ||  (is.na(list_val[i + 1])) == TRUE))
           {
             list_val[i + 1] <- rev_string3
           }
           else
           {
             list_val[i + 2] <- rev_string3
           }
         }
      }

    }
  }
return(list_val)

}
  if (str_detect(sequence, "GG") == FALSE)
  {
    print("The sequence inputed does not have a PAM sequence")
  }
}

seq1 <- protospacer(s1, l) #example output
seq1
```

```
## [1] "GATCTACTAGAGACTACTAA" "ATCTACTAGAGACTACTAAC" "TCTACTAGAGACTACTAACG"
```

```r
seq2 <- protospacer(s2, l)
seq2
```

```
## [1] "TAGTACATACTAACTCTAAC"
```

```r
seq3 <- protospacer(s3, 1)
```

```
## [1] "The sequence inputed does not have a PAM sequence"
```

```r
seq3
```

```
## [1] "The sequence inputed does not have a PAM sequence"
```